

---

**EEE8126: Computer Systems and Microprocessors**  
*Lab 2: C++(pointers, dynamic arrays, and call by references)*  
Module Instructor: Dr Rishad Shafik

---

**Exercise I: Introduction to C++ Pointers, and Call by Reference**

---

Recommended Time: 25 Mins Maximum

**Aims:**

- To understand the difference between C+ pointer variables and non-pointer variables
- To understand the difference between C++ call by value and call by reference

Follow the instructions below and try to do accordingly-

- DOWNLOAD** the source code of *pointer\_example2.cpp*.
- REVIEW** the source code of *pointer\_example2.cpp* using Notepad++ (Start->type "Notepad++")

Go through each line to understand how the code is organized. Check the following:

- Two non-pointer integer variables *B\_local* and *C\_local* are used together with two pointer integer variables *B\_pointer* and *C\_pointer*.
- The *B\_pointer* and *C\_pointer* are used to point to the addresses of *B\_local* and *C\_local*.
- The example uses two functions to add the *B\_local* and *C\_local* values: *add\_values\_cbv()* to add values through call by value, *add\_values\_cbr()* to add values through call by reference.
- The *add\_values\_cbv()* function copies the values of *B\_local* and *C\_Local* to the parameters *param\_B* and *param\_C*, while the *add\_values\_cbr()* function copies the memory references of *B\_local* and *C\_Local* through the pointer-type parameters *param\_B* and *param\_C*.

- COMPILE** the source code of *pointer\_example2.cpp*:
  - Start Cygwin command shell through Start->All Programs->Cygwin->Cygwin Bash Shell
  - In the Cygwin shell type: `g++ -Wall pointer_example2.cpp -o pointer_example`

The `-Wall` option enables all the warnings, and the `-o` option enables specification of the output executable. Your compilation should generate an executable called *pointer\_example*

- EXECUTE** the *pointer\_example* executable by typing the following in the Cygwin shell

```
./pointer_example
```

- OBSERVE** the outputs:
  - see how *B\_pointer* and *C\_pointer* can copy the memory references of *B\_local* and *C\_local*.

- see how values (not reference addresses) are copied by non-pointer variables within the *add\_values\_cbv()* function.
  - see how memory reference addresses are copied by pointer variables within the *add\_values\_cbr()* function
6. Now **UNCOMMENT** the lines 80-81 and 95-96 to enable local value modification within the two functions.
  7. **RECOMPILE, EXECUTE** and **OBSERVE** the following:
    - see how *B\_local* and *C\_local* values are unaffected by even after modification within the *add\_values\_cbv()* function
    - see how *B\_local* and *C\_local* values are affected by the modification within the *add\_values\_cbr()* function

**QUESTION:** Why did the *B\_local* and *C\_local* values change after the *add\_values\_cbr()* function? Why did the values not change after the *add\_values\_cbv()* function?

### **Exercise II: Call by Reference Using Arrays**

---

Recommended Time: 25 Mins Maximum Aims:

#### **Aims:**

- a. To understand how arrays can be used for modular operations through call by reference

Follow the instructions below and try to do accordingly-

9. **DOWNLOAD** the source code of *arrayreference.cpp*.
10. **REVIEW** the source code of *arrayreference.cpp* using Notepad++ (Start->type "Notepad++")

Go through each comment embedded and try to understand the program. Check the following:

- An double type array x is declared in the main function
  - When x is passed as an argument in functions, the first element address is passed as a call by reference
  - Hence, both *fill\_array()* and *change\_array()* functions can make amendments in the array elements, the first function fills the array with random double numbers and the second changes the values based on certain conditions (see the codes)
11. **COMPILE** the source code of *arrayreference.cpp* by
    - Start Cygwin command shell through Start->All Programs->Cygwin->Cygwin Bash Shell
    - In the Cygwin shell type: `g++ -Wall arrayreference.cpp -o arrayreference`

Your compilation should generate an executable called *arrayreference*

12. **EXECUTE** the *arrayreference* executable by

```
./arrayreference
```

13. **OBSERVE** the output and see how `x` elements are filled and changed using call by reference using `x` array as parameters in the functions

**QUESTION:** TRY to print the address of `x`'s first element within `main()` and also `x_param` address within the local function.

### **Exercise III: Review of Array based Call by Reference**

---

Recommended Tim: 45 Mins Maximum

#### **Aims:**

- a. To be able to write a C++ program using array based parameters
- b. To understand the call by reference using arrays in C++

14. **WRITE** a program (`encrypt.cpp`) to encrypt a message, which is an array of characters: `char msg[25]="Very Important Message"`, declared and initialized in the main function. The message will be encrypted as follows:

Update each character's ASCII code (i.e. `(unsigned int) msg[c]`, where `c` is the iterator) as  
Current ASCII code =  $((\text{previous ASCII code} * 9) \% 96) + 31$ ;

Implement the encryption through a function called `do_encrypt(char *)`. Before and also after this encrypt operation, show the encrypted message on the standard console output through another function, called `display_message(char *)`. Declare all the associated variables required.

### **Exercise IV: Dynamic Memory Allocation for Arrays**

---

Recommended Tim: 25 Mins Maximum

#### **Aims:**

- a. To be able to write a C++ program using dynamic memory allocations for arrays

15. **OBSERVE** the codes in `dynamic_array.cpp` and try to write the same for a random sized string or character array.

16. **TRY** to dump the output of the file in a CSV file, each five character followed by a line feed. Use `file_operations.cpp` as an example.

### **Annex: Take Home Exercise**

---

Read the history of substitution cipher here: [https://en.wikipedia.org/wiki/Substitution\\_cipher](https://en.wikipedia.org/wiki/Substitution_cipher)

Write a program that can read a large amount of text from an input file, encrypt using substitution cipher and save the outcome in a file. Use functions and call by references for each function.